CEWES MSRC/PET TR/98-24

VisGen Cross-Platform Visualization

by

M. Pauline Baker

Work funded by the DoD High Performance Computing
Modernization Program CEWES
Major Shared Resource Center through

Programming Environment and Training (PET)

Supported by Contract Number: DAHC 94-96-C0002
Nichols Research Corporation

VisGen Cross-Platform Visualization

M. Pauline Baker
NCSA
University of Illinois

March 1998

Abstract: In this focused effort, we are developing strategies that will allow us to build visualization tools for both the desktop and virtual reality display systems. We have taken care to encapsulate the functionality involved in mapping data to graphic form. This visualization-generator capability is then coupled with various user interfaces and used in various settings. To date, we have used our visualization generator in a desktop tool, and a Web-based VRML-server tool. We are working towards incorporation in an ImmersaDesk tool. In this setting, the VisGen code will be coupled with a multimodal interface, utilizing speech and gesture at the interface, and augmenting the visualization with non-speech audio.

1 Introduction

Virtual reality offers new functionality for visual data analysis. Stereoscopic viewing, multimodal interfaces, and high levels of interactivity all contribute to the experience of visualization in virtual reality. While the field is still quite immature, many prototype applications have been built. These are usually custom tools, often for a single application. Generally, they have little in common with desktop visualization tools that might be familiar to a scientist. Additionally, these tools take considerable time and effort to build and, once complete, usually run only on the virtual reality hardware.

In this report, we describe early efforts to build a visualization tool that executes on the desktop as well as on the ImmersaDesk. There are three issues that are significant in this endeavor: (1) how much of the visualization generation code can be reused?, (2) how to accommodate two very different display architectures?, and (3) how to support two very different user interfaces?

2 Visualization Generation

In this project, we isolated and encapsulated the functionality involved in reading data and generating graphical representations of that data. In particular, we were careful to insure that there were no dependencies between this functionality and the user interface or the display device.

Encapsulation of this kind of functionality might seem like an obvious thing to do. The well-known visualization package AVS adopted "modules" as a fundamental building block for the whole system. Each AVS module is specialized to do one particular job -- to read data, to filter the data in some way, to extract an isosurface, to calculate a particle trajectory, etc. There have been several attempts to use AVS on the ImmersaDesk or in the CAVE. This is seen as particularly desirable given the completeness of the AVS package. And, at first glance, the module structure would seem to lend itself well to providing a general-purpose visualization solution for virtual reality devices such as the ImmersaDesk. However, AVS made the design choice that each "module" includes the code to do its particular job, accompanied by the user interface code to manipulate the parameters associated with the task. That is, the isosurface module contains code to generate the surface polygons and code to draw and read from a user interface widget to set the isosurface threshold. Bundling the interface code along with the module's true task restricts the module's usefulness to the desktop screen, and limits its ability to transfer into the virtual reality arena.

In the VisGen effort, we have separated out the visualization generation functionality from any notion of user interface. We are leveraging the excellent work done on the freely available package, the Visualization ToolKit. We use VTK to implement the particular visualization algorithms that are of interest. VTK, like AVS, provides a large collection of visualization algorithms. However, VTK has a cleaner design in that user interface code is not bundled with the code to map data to geometry. This makes VTK an ideal choice for VisGen. The visualization generation functionality can be coupled with an appropriate interface and used in a variety of settings. To date, we have used the VisGen in three applications.

1. The most complete application is a traditional desktop tool. This tool reads the output of CE-QUAL-IQM and generates a variety of visual representations including isosurfaces, slices, volumes, etc. A beta version of this tool is currently in use by the Chesapeake Bay team at the CEWES MSRC. A preliminary User's Guide appears as Appendix A.

2. An emerging application for the VisGen is as the visualization generation functionality in a Virtual Reality tool for visual data analysis. It is premature to describe this use, except to say that different user interface techniques are required, and a different display method is used.

3. VisGen was also used in a prototype of a VRML-generator visualization tool. This type of visualization tool would be particularly useful to service the visualization needs of remote MSRC users. In this system, the researcher uses a Web form to indicate what data they were interested in

and to specify the visualization they needed. The Web form generates a request that is processed by a transaction broker. The transaction broker sends the user request to a VisGen instance. The VisGen services the request by generating and returning the visualization using VRML.

3 Targeted Displays

As described above, this effort aims to encapsulate visualization generation functionality such that the capabilities can be used across platforms. While we were initially interested in cross-platform use involving the desktop and the ImmersaDesk, we have found it useful to also consider VRML browsers as a target display.

In the desktop tool, we use a variety of rendering and display strategies. The VisGen can draw directly to the screen, using OpenGL. Alternatively, VisGen can capture a visualization as an image or a VRML file. This has been particularly useful for sharing visualizations on the Web, providing a form of asynchronous collaboration. We anticipate utilizing the same or similar functionality to support synchronous collaboration. We have done some experimentation with capturing the visualization as a RIB file. This is the standard file format used by RenderMan. Coupled with the RenderMan rendering system (we use the public domain version from Blue Moon), this provides a way to generate high-quality rendering of the visualization.

Virtual reality environments, such as the CAVE, are nicely supported by the graphics library Performer from SGI. VisGen can be used with Performer by building a translator that can convert from the VisGen/VTK geometry format to the Performer data format. As this strategy matures, it will be incorporated into the ImmersaDesk version of this set of cross-platform tools.

4 User Interfaces

As described above, the Web-accessible VRML-generator using VisGen relies only on Web forms for its user interface. In time, we will update this interface to use Java, which will provide for a richer, more complete interface. The desktop VisGen tool provides an easy-to-use, point-and-click interface, implemented using TCL/Tk.

It is the ImmersaDesk interface, as a virtual reality interface, that has the potential to offer the most interesting, multimodal user interface. The standard user interface device for the ImmersaDesk is the *wand* -- a hand-held, 6 DOF device, with 3 buttons and a thumb-activated joystick. For many applications, the wand is sufficient. For example, in architectural walkthroughs or other applications where the user simply travels through a virtual world, activating travel through a wand button and directing travel through the joystick's orientation works very well. The user interface for applications involving visual data analysis are far more complex, since the user must specify many parameters to define the visualization.

We have found a speech interface to be extremely helpful as an component in virtual reality applications. Until recently, speech interfaces were limited to systems that were speaker dependent, requiring training for each new user, and were often constrained to discrete words. Newer systems, commercially available and competitively priced, are speaker independent and support complex phrases and whole sentences. We are currently working with IBM's ViaVoice speech-recognition system. The developer kit available for this recognition engine supports the specification of a grammar defining the set of spoken strings comprising the speech interface. The grammar is a set of production rules. Alternatives are supported, so the grammar can be rich and natural-sounding. As part of this effort, we have written supporting software to make this functionality easy to use and embed in a virtual reality application. This software will be used in the ImmersaDesk VisGen.

We are also experimenting with using gesture to drive virtual reality applications. While gloves have been used in many virtual reality labs, we are beginning to use bare-hand gesture. This has the advantages of being free of all cords or other encumbrances. Cameras are mounted to the ImmersaDesk to record the user's hand and arm movements. Image-processing algorithms sort out the camera data stream and decode the user gesture. Currently, the "gesture" is of limited form -- position and orientation of the arm is traced and fed to the application. It is premature to think about using this technology to drive a complex visualization application, but bare-hand gesture will offer considerable advantages once it matures.

Appendix A

VizGen User's Guide

Version 1.0c <u>Draft</u> 1.0

Step-By-Step Tutorial

Code Developor: Robert Stein

Manual : Alan M. Shih

Sample Problem: Chesapeake Bay Water Quality Simulation
Performed by Dr. Carl Cerco et al. at Army Corp of Engineers Waterways Experiment Station

Contents:

Preparing Directories and Files

Step 1	Arrange the directories and files	Make sure that the data and files are in right place
Step 2	Check	Data/ directory is created under VizGen
Step 3	Check	Binaries/ directory is created under VizGen/Data/
Step 4	Check	All the executables are present in the directories as layout
Step 5	Check	The path on the first line of CreateBin.tcl and VizGen.tcl should be modified to the right path for "wish". Example path is set to be "#!/usr/local/bin/wish"

Converting Data

VizGen 1.0c requires the data to be converted into a binary format from wqm_apl.opt Following steps guide you through this painless process:

Step 1	Change directory to *VizGen/*
Step 2	Type **CreateBin.tcl**
Step 3	On the graphical user interface (GUI), select the property that you like to visualize, then press **Add**
Step 4	You can repeate Step 3 to add more properties onto the list at the right
Step 5	Once all the properties are all selected into the right-hand side, Press **Start**
Step 6	When all the data has been converted, a final staticstics will show up. Press **Quit**
Step 7	Check if all the binary files are created in VizGen/Data/Binaries

Basic Steps of VizGen

Step 1	Execute the program	Type "**go**" or "**cbay -f cbay.ini -gui VizGen.tcl**"
Step 2	Read data	Press "**Add**" under "*DataNodes*"
Step 3	Change to data directory	Double click on "*Data/*" and then "*Binaries/*" on the file selector
Step 4	Select data	Double click on the name of data file (e.g. salinity.bin)
Step 5	Create slice	Press "**Add**" under "*VizNodes*"
Step 6	Initial image shown	Click "**Slice**" and "**Salinity**" on the panel
Step 7	Access SLICE property panel	Double click the name "**Slice**" on the *Viznode* browser. (NOTE that the image will be reset to Z=0 at the first time)
Step 8	Change the location of the slice	Input or slide the value to the value needed.
Step 9	Change the MinMax of Data	Double click on the name of data (e.g. Salinity) in *DataNode* browser. Slides the MinMax values and press "**Apply**". Before and After.

Creating Animation

The following steps produce a movie file in QuickTime Movie format.

Step 1	Access the Movie output panel	Press the button Movie at the lower right corner
Step 2	Assign file name	Type a output file name, say, mymovie.mv
Step 3	Start recording	Press **Start**
Step 4	Start Animate	Press **Play**
Step 4	Stop recording	Press **Stop**
Step 5	Review recorded file	* Type "**movieplayer** mymovie.mv" or * Go to your Web browser, open the file mymovie.mv (Note: You need to have proper plug-in in your Web brower to view the movie). Check it out with this clip!